Robust Three-View Triangulation Done Fast

Johan Hedborg, Andreas Robinson and Michael Felsberg

Hedborg, J., Robinson, A., Felsberg, M., (2014), Robust Three-View Triangulation Done Fast, *Proceedings: 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPRW 2014*, , 152-157. https://doi.org/10.1109/CVPRW.2014.28

Original publication available at:
https://doi.org/10.1109/CVPRW.2014.28

Robust Three-view Triangulation Done Fast

Johan Hedborg
Linköping University
johan.hedborg@liu.se

Andreas Robinson
Linköping University
andro134@student.liu.se

Michael Felsberg
Linköping University
michael.felsberg@liu.se

Abstract

Estimating the position of a 3-dimensional world point given its 2-dimensional projections in a set of images is a key component in numerous computer vision systems. There are several methods dealing with this problem, ranging from sub-optimal, linear least square triangulation in two views, to finding the world point that minimized the L_2-reprojection error in three views. This leads to the statistically optimal estimate under the assumption of Gaussian noise. In this paper we present a solution to the optimal triangulation in three views.

The standard approach for solving the three-view triangulation problem is to find a closed-form solution. In contrast to this, we propose a new method based on an iterative scheme. The method is rigorously tested on both synthetic and real image data with corresponding ground truth, on a midrange desktop PC and a Raspberry Pi, a low-end mobile platform.

We are able to improve the precision achieved by the closed-form solvers and reach a speed-up of two orders of magnitude compared to the current state-of-the-art solver. In numbers, this amounts to around 300K triangulations per second on the PC and 30K triangulations per second on Raspberry Pi.

1. Introduction

Triangulation, i.e. finding the three-dimensional location of a point observed from two or more viewpoints, is a fundamental problem in computer vision. Over the years, many methods have been proposed, in particular for the case of two views, where robust and fast methods have been developed [1, 2].

It is well known that adding one more view significantly improves the estimate, but this also increases the complexity of the problem [3]. Still, there is a closed-form analytical solution to the three-view triangulation problem, where theory guarantees that a solution exits. In practice, however, this solver sometimes fails due to limitations in floating point precision.

In this paper, we state the hypothesis that by replacing the closed form solver with an iterative approach, we simultaneously reduce the failure rate and the computational effort. We confirm this hypothesis in a number of experiments, including testing it on the popular Raspberry Pi, a small-form-factor computer based on a system-on-chip for mobile applications.

2. Related work

Hartley and Sturm [4] presented a non-iterative solver of the two view problem, guaranteed to find a global optimum named the *polynomial method*. Performing optimal triangulation, it minimizes the L_2 norm of the error between observed and reprojected 2D points while additionally enforcing the epipolar constraint. This is more robust than older methods such as the Linear-Least-Squares (LLS) but comes with a cost of 7 - 15 times longer running time.

Kanatani et al. [5] found that the polynomial method has singularities in the epipoles and propose an iterative "optimal correction" method that avoids the singularities by approaching the epipolar constraint, only satisfying it on the last iteration. Their method returns identical results to those of Hartley and Sturm [4] while being much faster.

However, Lindstrom [6] points out that Kanatani's method [5] may only find a local minimum and that the number of iterations required is its main drawback. He develops a pair of methods that take provably optimal steps in each iteration such that the intermediate sub-optimal result always fulfils the epipolar constraint, permitting early termination. For the vast majority of points, convergence occurs in two iterations allowing the methods to be reformulated as non-iterative. Lindstrom reports at least eight-digit agreement with [4] of the reprojection errors on synthetic datasets and between three and seven digits on real-world datasets. In terms of speed, Lindstrom [6] is 50 times faster than the polynomial method and 20 times faster than the method of Kanatani et al..

Where the globally optimal polynomial method of Hartley and Sturm [4] finds six roots of a polynomial equation seeking to minimize the L_2-reprojection error in two views, there are a set of methods using the same error metric but

in three views instead. Stewénius et al. [3] find 47 roots using the Gröbner basis method to locate the optimal solution for three views. This method is however costly as a 128-bit mantissa is needed for stability, resulting in long run times.

Byröd et al. [7] improve the numerical stability of Stewenius method with what they call the "relaxed ideal" and remove the need for high-precision floating point computations. On the real-world Oxford "Dinosaur" turntable sequence, the method is 480 times faster (2.5 minutes vs 20 hours) while also exhibiting 10^{-5} of the error of Stewenius method on synthetic data.

The same authors further improve the numerical stability of this result by adapting the Gröbner solver to the specific problem at hand, by changing basis of the quotient space, found through singular-value decomposition [8]. As the SVD is computationally expensive and dominates the algorithm, they further suggest to reduce the size of the basis-finding problem using QR-factorization with column pivoting [9]. This modification is found to be four times faster than without QR factorization while maintaining robustness.

Kukelova et al. [2] propose using Lagrange multipliers to search for the global optimum, generating a system of eight polynomial equations with 31 roots, for which the authors develop a specialized Gröbner basis solver. The resulting method is more than five times faster than the QR based method on real-world datasets, more robust and as accurate on synthetic data.

Finally, Nordberg [1] develops a radically different approach by first finding the triangulation tensor $K \in \mathbb{R}^{4\times 27}$ describing the spatial relationships between three cameras and then generating a tensor product Y for each triplet of homogeneous 2D points. The 3D coordinate is subsequently found directly from a matrix-vector multiplication $X = KY$. This is fast, but constructing and optimizing K from camera matrices adds a setup cost to the method.

In this paper we also address the three-view triangulation problem. The problem formulation is the same as the one solved in Byröd et al. and Kukelova et al., where the method proposed by Kukelova et al. is the currently fastest solver. Our contribution is a method that is two orders of magnitude faster than the current state-of-the-art solver by Kukelova et al. [2].

3. Method

3.1. Problem Formulation

The problem is stated as follows; given three projection matrices $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3 \in \mathbb{R}^{3\times 4}$, and three image projections $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \in \mathbb{R}^2$ of a scene point $\mathbf{w} \in \mathbb{R}^3$ and with $\mathbf{w}^h \in \mathbb{R}^4$ being the homogenous representation of $\mathbf{w}$. The image residual for one re-projection i is defined as

$$\mathbf{r}_i = \mathbf{x}_i - \text{proj}(\mathbf{P}_i \mathbf{w}^h) \tag{1}$$

where the projection mapping proj() $\in \mathbb{R}^3 \rightarrow \mathbb{R}^2$ is defined as

$$\text{proj}(\mathbf{y}) = [\frac{y_1}{y_3} \;\; \frac{y_2}{y_3}]^T. \tag{2}$$

We want to find the position of the 3D point $\mathbf{w}$ such that

$$f = ||\mathbf{r}||_2^2, \quad \mathbf{r} = [r_{1x}\; r_{1y}\; ...\; r_{3x}\; r_{3y}]^T \tag{3}$$

is minimized. When the error metric is defined as the L_2 error in the image plane, as in this case, the triangulation method is known as an optimal method [4]

3.2. Non-linear optimization

Minimizing (3) is a non-linear least squares problem, and can be solved by e.g. the Gauss-Newton (GN) method [10] or the Levenberg-Marquart (LM) algorithm, which combines GN and steepest descent (SD) using a damping factor. We have instead chosen to apply an alternative method that combines GN and SD by explicitly using the radius of a trust region, called *Dog Leg* (DL) [11], because "The Dog Leg method is presently considered as the best method for solving systems of non-linear equations." [12]. Lourakis and Argyros [13] have shown that DL performs bundle-adjustment 2 to 7.5 times faster than sparse LM while maintaining equivalent quality.

The combination of GN and SD is determined by means of the radius Δ of a *trust region*. Within this trust region, we assume that we can model $\mathbf{r}(\mathbf{w}) \in \mathbb{R}^3 \rightarrow \mathbb{R}^6$ locally using a linear model $\boldsymbol{\ell}$:

$$\mathbf{r}(\mathbf{w}+\mathbf{h}) \approx \boldsymbol{\ell}(\mathbf{h}) \triangleq \mathbf{r}(\mathbf{w}) + \mathbf{J}(\mathbf{w})\mathbf{h} \;\; , \tag{4}$$

where $(\mathbf{J}(\mathbf{w}))_{ij} = \frac{\partial \mathbf{r}_i}{\partial w_j}(\mathbf{w})$ is the Jacobian. Each of the three 2×3 blocks (index i) in the Jacobian reads

$$\frac{\partial \mathbf{r}_i(\mathbf{w})}{\partial \mathbf{w}} = -\frac{\partial \text{proj}(\mathbf{y})}{\partial \mathbf{y}} \frac{\partial \mathbf{y}(\mathbf{w})}{\partial \mathbf{w}}, \;\; \mathbf{y}(\mathbf{w}) = \mathbf{P}_i \mathbf{w}^h. \tag{5}$$

Applying the chain rule in this way, does not only give the simplest solution, but is also beneficial in terms of speed.

The GN update $\mathbf{h}_{\text{GN}}$ is obtained by solving $\mathbf{J}(\mathbf{w})\mathbf{h} = -\mathbf{r}(\mathbf{w})$ e.g. by Gaussian elimination with pivoting.

If the update is within the trust region ($\|\mathbf{h}_{\text{GN}}\|_2 \leq \Delta$), it is used to compute a new potential parameter vector

$$\mathbf{w}_{\text{new}} = \mathbf{w} + \mathbf{h}_{\text{GN}} \;\; . \tag{6}$$

Otherwise, compute the gradient $\mathbf{g} = \mathbf{J}^T\mathbf{r}$ and the steepest descent step $\mathbf{h}_{\text{SD}} = -\alpha\mathbf{g}$ (for the computation of the step length α, see [14]). If the SD step leads outside the trust region ($\alpha\|\mathbf{g}\|_2 \geq \Delta$), a step in the direction of steepest descent with length Δ is applied

$$\mathbf{w}_{\text{new}} = \mathbf{w} - (\Delta/\|\mathbf{g}\|_2)\mathbf{g} \;\; . \tag{7}$$

If the SD step is shorter than Δ, β times $\mathbf{h}_{\mathrm{GN}} + \alpha\mathbf{g}$ is added to produce a vector of length Δ:

$$\mathbf{w}_{\mathrm{new}} = \mathbf{w} - \alpha\mathbf{g} + (\mathbf{h}_{\mathrm{GN}} + \alpha\mathbf{g})\beta \ . \tag{8}$$

In all three cases, the new parameter vector is only used in the next iteration ($\mathbf{w} = \mathbf{w}_{\mathrm{new}}$) if the gain factor ρ is positive (for the computation of ρ see [14]). Depending on the gain factor, the region of trust is growing or shrinking. The iterations are stopped, if either of $\|\mathbf{g}\|_\infty$, $\|\mathbf{r}\|_\infty$, $\|\mathbf{h}\|_2$, or Δ is below a threshold or if a maximum number of iterations is reached.

3.3. Initialization

As in all iterative solutions to non-linear optimization problems, a good initialization is needed. For some problems e.g. a bundle adjustment problem [15], the initialization is non-trivial and usually requires a number of methods, in order to have a good chance of convergence. Fortunately, in the case of initialization of our solver, there are very fast and quite robust two-view methods that are perfectly suited. The mid-point two-view triangulation method falls into this category and provides us with a good initial guess. Due to the simplicity of the mid-point method, it is numerically stable, and even if the result is not satisfactory for a final triangulation, it is useful for initialization.

We will thoroughly demonstrate this by showing that the proposed non-linear optimization approach finds the solution for significantly more cases than the non-iterative method of Byröd et al.

4. Evaluation

In order to prove our hypotheses we have evaluated the method of Hartley and Sturm, Byröd et al., and the proposed, on 6 datasets, three of which are synthetic and three of which are from real image data. The synthetic datasets are used to illustrate that for fairly well conditioned data all methods show good performance, while for data that is less well behaved, e.g. for a forward moving camera, errors are introduced.

4.1. Synthetic Dataset

Synthetic data was generated as follows: A 4000-point 3D point cloud uniformly distributed in the shape of a cube, with the size of $[-0.5, 0.5]^3$, was projected into a set of 40 pinhole cameras. The field of view angle is 46° and the image size is 1280×720. These 2D projections were then perturbed with noise distributed as $\mathcal{N}(0, \sigma)$, $\sigma = 1$ pixel.

Camera trajectories created in three ways: (1) "orbital" - a circle motion around the center of the point cloud, at a distance of 2. The trajectory ranges from -0.5 to 0.5 radians. (2) "lateral" - positioned on a straight line, facing the point could, and with a distance of 2 from the center. The lateral range is $[-0.5, 0.5]$. (3) "forward" - camera moves along a line towards the center of the point cloud. The camera is facing forwards and the trajectory range is $[-1.5, -0.5]$. The camera trajectories are slightly perturbed with a uniform distribution, $\mathcal{U}(0, 0.01)$ for the position and $\mathcal{U}(0, 0.01)$ radians for the rotation. 3D visualizations of the respective datasets are shown in the left column, first three rows, in figure 2.

4.2. Real Dataset

The real world datasets used are (1) the Oxford "dinosaur" [16], (2) the photo-tourism [17] "Notre Dame" and (3) the Oxford "corridor" [16]. All datasets, except for the dinosaur, had an associated point cloud that was refined using bundle adjustment. The refined point cloud was used as ground truth. In accordance with the rest of the real datasets, the ground truth for the dinosaur dataset was created by a bundle adjustment refinement. The three different datasets are visualized in the left column, three bottom rows, in figure 2.

4.3. Experimental Setup

We test three methods: The Polynomial method of Hartley and Sturm, the QR-method by Byröd et al. and the one presented here. The original code for the method of Byröd et al. is available online. We were not able to compare precision between our method and the method of Kukelova et al. because their code is currently not available. However we can compare the speed as it is stated in their paper. For the proposed method we have limited the number of iterations to 10, this gave us convergence in almost all cases.

To reconstruct a 3D point, the triangulation function receives a triplet of matching 2D points and corresponding cameras. The 2D point correspondences were chosen as the first, middle, and last point of its respective trajectory. This will not result in the widest baseline for all points, however as our intent is to compare the performance of the methods with one another and not absolutely, the exact choice is not crucial. Note that in the evaluation we have chosen the same set of point correspondences for all three methods.

Accuracy is evaluated by computing the distributions of two error measures; the 3D estimation error $\|\hat{\mathbf{w}} - \mathbf{w}\|_2$ and the re-projection error $\|\hat{\mathbf{x}} - \mathbf{x}\|_2$, where $\mathbf{w}$ are ground truth points and $\hat{\mathbf{x}}$ are the reprojected points of $\hat{\mathbf{w}}$ The results are summarized in figure 2.

5. Results

5.1. Comparative Experiment

As expected, the three-point methods have lower errors than the two-point Polynomial method. Furthermore, ours shows good agreement with Byröd's method, in some cases to the point where it is hard to separate the two graphs from

each other. Interestingly, Byröd's method fails to correctly triangulate some points that both our and the Polynomial method can deal with. This is apparent in the 3D error histogram of the Notre Dame dataset as a small bump around 10^0 and similarly in the Forward and Dinosaur sets. In the corresponding reprojection error histograms, these incorrectly triangulated points gather in the rightmost bin. It is possible that these errors are a product of the elimination of base vectors in the method.

5.2. Numerical Stability

All methods were run on a noise free version of the synthetic forward dataset, in order to evaluate the numerical stability for each of the methods. The result is shown in figure 1. Here we can see that the stability of the proposed method is significantly better than of the competing non-iterative thee-view method. The two-view optimal method is expected to perform well here due to the absence of noise and the lower computational complexity of the solver. When noise is introduced to the data both three-view solvers will surpass it, as can be seen in figure 2.

In the paper by Kukelova et al [2] there are similar graphs as in figure 1 showing that their method has better accuracy than Byröd et al[9]. However they do not show such a large difference in the precision as can bee seen in figure 1.

5.3. Speedup

The table below shows the average runtime per triangulation on each of the datasets, when executed on a Xeon W3550 @ 3.07 GHz.

Dataset	PC
Orbital	3.0 μs
Lateral	2.9 μs
Forward	3.8 μs
Dinosaur	2.9 μs
Notre Dame	2.6 μs
Corridor	3.4 μs

Our method is implemented in C++ while the QR-method is written in Matlab's interpreted code. To assess and reduce the impact of this difference, some extra steps had to be taken. Using Matlab's profiler it was determined that a small set of built-in linear algebra functions (`eig`, `qr`, `triu` and `lu`, found on lines 94, 97 and 132 in `tvt_solve_qr.m`, matlab source code released with [9]) account for 72.4% of the total runtime.

Matlab's computational backend for linear algebra operations is the Math Kernel Library[1], designed by Intel to be very fast[18]. It is therefore likely that these Matlab functions are as fast as any equivalent C++ implementation.

[1]This can be verified by typing as "version -blas" or "version -lapack" on the Matlab command line.

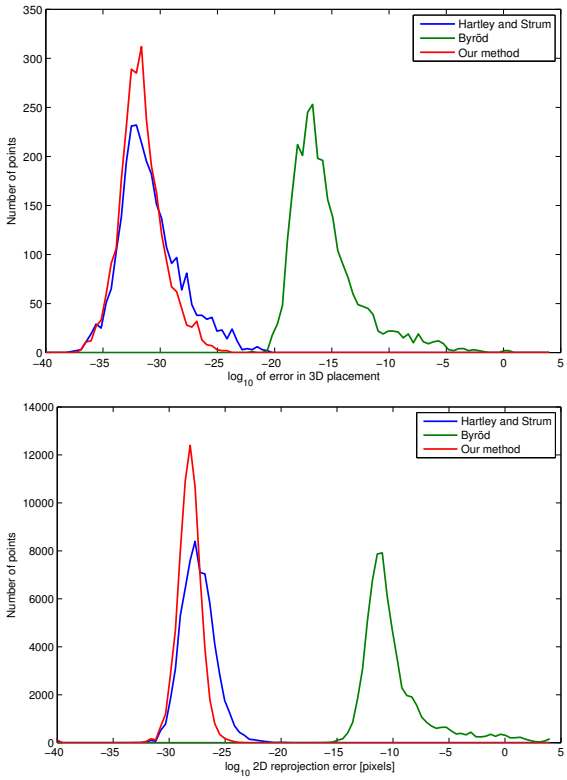

Figure 1. Triangulation errors for the Forward dataset without noise. Top: 3D reconstruction error histograms, bottom: reprojection error histograms.

The execution time for Byröds method was thus conservatively defined as the total execution time of these three lines of code, amounting to 5.14 ms.

The method of Kukelova et al. could not be timed, as the code is unavailable. However [2] states a runtime of 1 ms.

On the PC, the mean execution time per triangulation for the proposed method is 3.1 μs, 320 times faster than the three-view solver of Kukelova et al and 1600 times faster than the method of Byröd et al.

On the Raspberry Pi we run the Notre dame data set and had a execution time of 31 us per triangulation. This makes a Raspberry Pi, utilizing the proposed method, triangulate 30 times faster than a modern PC running the method of Kukelova et al.

6. Conclusion

We have presented a solver for the three-view triangulation problem, based on the Dog Leg iterative non-linear least squares solver. The method is over 300 times faster than the fastest state-of-the-art non-iterative method on a PC, and more than 30 times faster on a Raspberry Pi, while still achieving better precision. The Notre Dame data set of 127431 points is triangulated in 0.26 seconds on the PC and in 3.7 seconds on the Pi.

Acknowledgments This work has been supported by ELLIIT, the Strategic Area for ICT research, funded by the Swedish Government, VPS, funded by the Swedish Foundation for Strategic Research, and Extended Target Tracking (within the Linnaeus environment CADICS), from the ECs 7th Framework Programme (FP7/2007-2013).

References

[1] K. Nordberg, "Efficient three-view triangulation based on 3d optimization," in *British Machine Vision Conference (BMVC)*, 2008. 1, 2

[2] Z. Kukelova, T. Pajdla, and M. Bujnak, "Fast and stable algebraic solution to l2 three-view triangulation," in *3DTV-Conference, 2013 International Conference on*. IEEE, 2013, pp. 326–333. 1, 2, 4

[3] H. Stewenius, F. Schaffalitzky, and D. Nister, "How hard is 3-view triangulation really?" in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 1. IEEE, 2005, pp. 686–693. 1, 2

[4] R. I. Hartley and P. Sturm, "Triangulation," *Computer vision and image understanding*, vol. 68, no. 2, pp. 146–157, 1997. 1, 2

[5] K. Kanatani, Y. Sugaya, and H. Niitsuma, "Triangulation from two views revisited: Hartley-sturm vs. optimal correction," *In practice*, vol. 4, p. 5, 2008. 1

[6] P. Lindstrom, "Triangulation made easy," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 1554–1561. 1

[7] M. Byröd, K. Josephson, and K. Åström, "Fast optimal three view triangulation," in *Computer Vision–ACCV 2007*. Springer, 2007, pp. 549–559. 2

[8] M. Byrod, K. Josephson, and K. Astrom, "Improving numerical accuracy of gröbner basis polynomial equation solvers," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, 2007, pp. 1–8. 2

[9] M. Byröd, K. Josephson, and K. Åström, "A column-pivoting based strategy for monomial ordering in numerical gröbner basis calculations," in *Computer Vision–ECCV 2008*. Springer, 2008, pp. 130–143. 2, 4

[10] M. Sarkis, K. Diepold, and K. Huper, "A fast and robust solution to the five-pint relative pose problem using gauss-newton optimization on a manifold," in *Acoustics, Speech and Signal Processing, 2007*, 2007. 2

[11] M. J. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The computer journal*, vol. 7, no. 2, pp. 155–162, 1964. 2

[12] K. Madsen, H. Nielsen, and O. Tingleff, "Methods for non-linear least squares problems," IMM, Technical University of Denmark, Tech. Rep., April 2004, 2nd Edition. 2

[13] M. Lourakis and A. A. Argyros, "Is levenberg-marquardt the most efficient optimization algorithm for implementing bundle adjustment?" in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2. IEEE, 2005, pp. 1526–1531. 2

[14] J. Hedborg and M. Felsberg, "Fast Iterative Five point Relative Pose Estimation," in *IEEE Workshop on Robot Vision (WoRV 2013), January 15-17, 2013, Clearwater, FL, USA*. IEEE conference proceedings, 2013. 2, 3

[15] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment a modern synthesis," in *Vision Algorithms: Theory and Practice*, ser. Lecture Notes in Computer Science, B. Triggs, A. Zisserman, and R. Szeliski, Eds. Springer Berlin Heidelberg, 2000, vol. 1883, pp. 298–372. 3

[16] Multi-view data-sets. [Online]. Available: http://www.robots.ox.ac.uk/~vgg/data1.html 3

[17] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3d," in *SIGGRAPH Conference Proceedings*. New York, NY, USA: ACM Press, 2006, pp. 835–846. 3

[18] Intel math kernel library. [Online]. Available: http://software.intel.com/en-us/intel-mkl 4

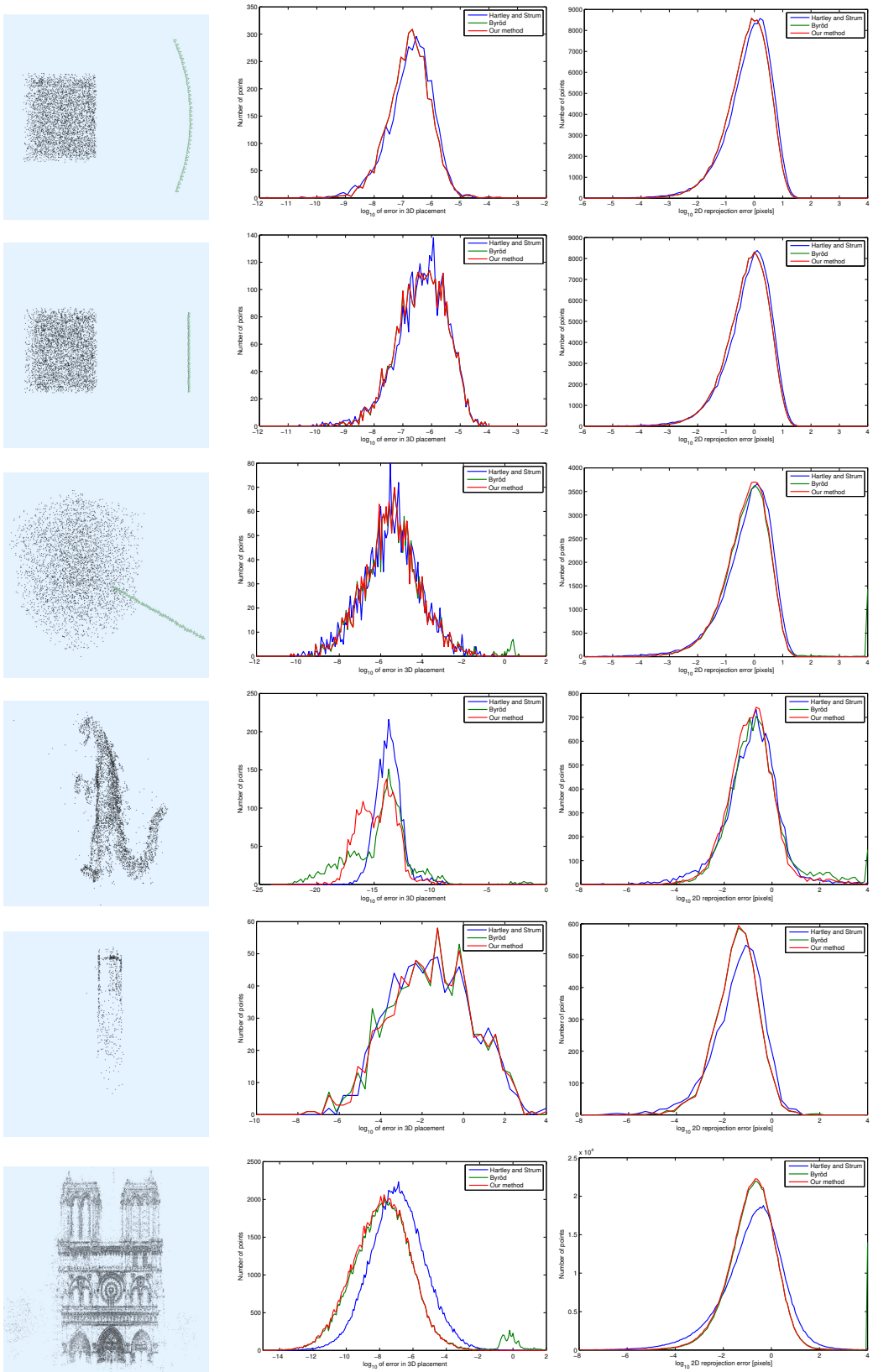

Figure 2. Triangulation errors. Rows (top to bottom): Orbital, Lateral, Forward, Dinosaur, Corridor, and Notre Dame datasets. Columns (left to right): Point cloud visualization (with cameras only shown in synthetic sets), 3D reconstruction error histograms, and reprojection error histograms. Preferably viewed in color.